artdaq Utilities - Idea #14606

Implement the transition scripts called by sdaqinterface.py for high-level control of artdaq processes

11/22/2016 05:22 PM - John Freeman

Status:	Resolved	**Start date:**	11/22/2016
Priority:	Normal	**Due date:**	
Assignee:	John Freeman	**% Done:**	100%
Category:	artdaq-daqinterface	**Estimated time:**	60.00 hours
Target version:		**Spent time:**	0.00 hour
Experiment:	-		

Description

Recently, in artdaq-utilities-sdaqinterface (http://cdcvs.fnal.gov/projects/artdaq-utilities-sdaqinterface), a "simple daqinterface" script, sdaqinterface.py, was implemented. This script respects a set of state transitions, where for each transition, it calls a bash script in charge of that transition. The actual implementation of the bash scripts is left to the user of sdaqinterface. This Redmine Issue describes the need for a general-purpose implementation of the scripts, so that, once an experiment has its hands on the scripts, it should be relatively straightforward to get them to work for a specific experiment's artdaq-based DAQ system. Along with the scripts themselves, documentation on how to use them should additionally be provided.

History

#1 - 11/22/2016 05:30 PM - John Freeman

- % Done changed from 0 to 20

Work has already been done on this project; as of this writing it can be found on github, https://github.com/jcfreeman2/DAQInterface.git , though we should think about where to give it a permanent home. It's basically a modification of the DAQInterface python code used on the 35ton experiment; an email describing in detail my plans for it was sent to artdaq-developers@fnal.gov yesterday (Nov. 21, 2016):

```
Hi all,

    In light of our recent discussions concerning the creation of an LRP (Long Running Process) through which to control artdaq processes, I've
begun overhauling the 35ton DAQInterface python script so as to support this effort. Essentially, it will serve as an intermediary between Wes's
sdaqinterface.py script (found in http://cdcvs.fnal.gov/projects/artdaq-utilities-sdaqinterface) and the artdaq processes. To get more specific, as of
commit b49c0fac51e6972876c562f56d581f0bc130cf5c (dated Oct. 19) sdaqinterface.py supports a state machine model of the following form:
Existing -> "Initialize" -> Initialized -> "Boot" -> Booted -> "Config" -> Configured -> "Run" -> Running -> "Stop" -> Stopped -> "Shutdown" ->
Initialized -> "Terminate" -> NO STATE. This state machine doesn't match up 1-1 with the state machine supported in artdaq v1_13_03, so
DAQInterface takes account of this, accepting the sdaqinterface transitions via XML-RPC and "translating" them into artdaq transitions. How this
is done, I describe below. If you have any questions / concerns / disagreements, please let me know. For each sdaqinterface transition, I describe
things in the following format:

"Transition"
Argument(s): # of arguments
-Description of argument #1, etc.

<<Starting State #1 -> Final State>>
Steps taken by DAQInterface when issuing transition "Transition" from Starting State #1

<<Starting State #2 -> Final State>>
Steps taken by DAQInterface when issuing transition "Transition" from Starting State #2

...etc...

Let's begin.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

"Initialize"
```

```
Argument(s): 0

<<Existing -> Initialized>>:
Launch DAQInterface (error/warning if it already exists?)

<<Initialized -> Initialized>>:
No-op (unless it doesn't exist, in which case issue warning and then launch DAQInterface)

////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

"Boot"
Argument(s): 2+
-DAQInterface configuration file (provides # of eventbuilders and # of aggregators, location of artdaq-demo bu
ild)
-Component list (names of fragment generators)

<<Initialized -> Booted>>:
Create the eventbuilders, aggregators and boardreaders via pmt.rb

////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

"Configure"
Argument(s): 2
-Name of configuration in database
-Name of subdirectory associated with configuration

<<Booted -> Configured>>
-Take the FHiCL documents from the configuration
-Adjust them for bookkeeping purposes (based on # of eventbuilders, etc.)
-Send the artdaq "init" transition w/ the FHiCL documents

<<Configured -> Configured>>
-Repeat the steps from the <<Booted -> Configured>> case

<<Stopped -> Configured>>
-Send the artdaq "shutdown" transition
-Repeat the steps from the <<Booted -> Configured>> case

////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

"Run"
Argument(s): 1
-Run #

<<Configured -> Running>>
-Send the artdaq "start" transition w/ the run #
-Execute optional experiment-specific "experiment_run()" function (unless overridden, this is a no-op)

<<Stopped -> Running>>
-Repeat the steps from the <<Configured -> Running>> case

////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

"Stop"
Argument(s): 0

<<Running -> Stopped>>
-Execute optional experiment-specific "experiment_stop()" function (unless overridden, this is a no-op)
-Send the artdaq "stop" transition

////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

"Shutdown"
Argument(s): 0

<<Stopped -> Initialized>>
-Send the artdaq "shutdown" transition
-Kill the artdaq processes

<<Configured -> Initialized>>
```

```
-Same steps as <<Stopped -> Initialized>>

<<Booted -> Initialized>>
-Kill the artdaq processes

////////////////////////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////

"Terminate"
Argument(s): 0

<<ANY STATE -> Terminated>>
-If in running state, make every effort to close output file and shut down hardware cleanly (i.e., attempt sto
p transition, etc.)
-Kill artdaq processes
-Kill DAQInterface
```

#2 - 12/20/2016 05:21 PM - John Freeman

- Status changed from New to Resolved

- % Done changed from 20 to 100

The needed SDAQ_*.sh scripts have been added to artdaq-utilities-daqinterface (http://cdcvs.fnal.gov/projects/artdaq-utilities-daqinterface) via commit 23a67b638a8fdb58e243b8436ea31dedbefbf783 , and I've been able to control DAQInterface by running the rundaq.py script from the commit b49c0fac51e6972876c562f56d581f0bc130cf5c version of artdaq-utilities-sdaqinterface after commenting/uncommenting the relevant lines in rundaq.py so that it uses the cmdlineControlInput module for input (i.e., takes commands at the command line).

#3 - 01/13/2017 04:41 PM - Eric Flumerfelt

- Project changed from artdaq to artdaq Utilities

- Category set to 430

#4 - 02/10/2017 01:07 PM - Eric Flumerfelt

- Category changed from artdaq-utilities to artdaq-daqinterface